YAMAHA
SR1 プロバイダ

Version 1.0.0

ユーザーズ ガイド

October 31, 2012

備考:

【ご注意】

本プロバイダを使用する場合は別途“SR1 プロバイダ”ライセンスが必要です.

目次

1. はじめに

本書は YAMAHA ロボット SR1/DRCX シリーズ 単軸/2 軸用の CAO プロバイダのユーザーズガイドです. 本書で扱う CAO プロバイダ(CaoProvSR1.dll)を SR1 プロバイダと呼びます.

次章に SR1 プロバイダの概要, 3 章にコマンドリファレンスを記載しています.

1.1. 非常停止スイッチの設置

ロボットコントローラを使用になる前に, 非常の際にただちにロボットの運転を停止できるよう, 作業者が容易に操作できる位置に非常停止スイッチを設置してください.

(1) 非常停止スイッチは, 赤色にしてください.

(2) 非常停止の機能は, 作動させたあと自動的に復帰せず, また他の作業者が不用意に復帰させることができないようにしてください.

(3) 非常停止スイッチは, 電源スイッチとは別個に設けてください.

2. プロバイダの概要

2.1. 概要

SR1 プロバイダは，YAMAHA ロボットコントローラに依存する部分を吸収し CAO プロバイダ・インターフェース仕様で規定された機能を提供する CAO プロバイダです．そのファイル形式は DLL(Dynamic Link Library)であり，CAO エンジンから使用時に動的にロードされます．SR1 プロバイダを使用するにあたっては ORiN2SDK をインストールするか，下表を参照して手作業でレジストリ登録を行う必要があります．

表 2-1 SR1 プロバイダ

ファイル名	CaoProvSR1.dll
ProgID	CaoProv.YAMAHA.SR1
レジストリ登録	regsvr32 CaoProvSR1.dll
レジストリ登録の抹消	regsvr32 /u CaoProvSR1.dll

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddControllerメソッド

SR1 プロバイダでは AddController 時に通信用の接続パラメータを参照し，通信の接続を行います．このときオプションで通信形態，タイムアウト，Ethernet 通信の場合はユーザ名，パスワード，Telnet のポート番号を指定します．

書式 AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPCName:BSTR>,<bstrOption:BSTR>)

以下にオプション文字列に指定するリストを示す．

表 2-2 CaoWorkspace::AddController のオプション文字列

オプション	意味
Conn=<接続パラメータ>	必須．通信形態とその接続パラメータを設定します．
[User=<ユーザ名>]	YAMAHA コントローラにログインするためのユーザ名を指定します．(デフォルト:admin)
[Password=<パスワード>]	ログインするためのパスワードを指定します．(デフォルト:無し)
[Timeout=<タイムアウト時間>]	送受信時のタイムアウト時間(ミリ秒)を指定します．(デフォルト:500)

2.2.1.1. Connオプション

以下に Conn オプションの接続パラメータ文字列を示します．ここで角括弧(“[]”)内のパラメータは省略可能を示します．また，各パラメータの解説中の下線部はオプション指定を省略した時のデフォルト値を示します．

・ **Ethernet** デバイス

“Conn=eth:<IP Address>[:<PortNo>]”

<IP Address> : 必須． 接続先の IP アドレス．
例 : ”127.0.0.1”, ”192.168.0.1”

<PortNo> : 接続先のポート番号．
例 : “127.0.0.1:<u>23</u>”, ”192.168.0.1:5010”

・ **RS232C** デバイス

“Conn=com:[<ComPort>”[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>]]]”

<ComPort> : COM ポート番号．’<u>1</u>’-COM1, ’2’-COM2,...

<BaudRate> : 通信速度．4800, 9600, 19200, <u>38400</u>, 57600, 115200

<ByteSize> : パリティ. 'N'-NONE, 'E'-EVEN, 'O'-ODD
<DataBits> : データビット数. '7'-7bit, '8' – 8bit
<StopBits> : ストップビット数. '1'-1bit, '2'-2bit

2.2.2. CaoController::AddRobotメソッド

CaoController クラスの AddRobot メソッドの引数は, ロボット名(BSTR 型)を指定します. ここで指定するロボット名は任意の文字列で指定することができます. AddRobot メソッドを呼び出すと CaoRobot オブジェクトが取得できます.

書式 AddRobot(<bstrName:BSTR>[,<bstrOption:BSTR>])

< bstrName > : [in] ロボット名(VT_BSTR)
< bstrOption > : [in] オプション文字列(未使用)

2.2.3. CaoController::AddVariableメソッド

CaoController クラスの AddVariable メソッドは, 変数にアクセスするためのメソッドです. SR1 プロバイダでは, 変数名にシステム変数を指定します.

SR1 プロバイダで実装されているシステム変数は表 2-5を参照してください.

書式 AddVariable(<bstrVariableName:VT_BSTR>[,<vntOption:VT_BSTR>])

< bstrVariableName > : [in] 変数名(VT_BSTR)
<bstrOption> : [in] オプション文字列

使用例

```
Dim aaa As Object
Dim bbb As Double

Set aaa = caoCtrl.AddVariable("@POS")
bbb = aaa.Value
```

2.2.4. CaoController::Executeメソッド

コマンドを実行します.

Execute メソッドの引数は, コマンドを BSTR,パラメータを VARIANT 配列で指定します.

各コマンドの詳細は3コマンドリファレンスを参照してください.

書式 [<vntRet:VT_VARIANT>=]Execute(<bstrCmd:VT_BSTR>[,<vntParam:VT_VARIANT>])

< vntRet > : [out] コマンドの返り値 (VT_VARIANT)
< bstrCmd > : [in] コマンド (VT_BSTR)
< vntParam > : [in] パラメータ (VT_VARIANT)

使用例

```
Dim vRes As Variant
vRes = caoCtrl.Execute("SRVO", 1)  ' サーボを ON
```

2.2.5. CaoRobot::AddVariableメソッド

CaoRobotクラスのAddVariableメソッドは，変数にアクセスするためのメソッドです．SR1プロバイダでは，変数名にシステム変数を指定します．

SR1 プロバイダで実装されているシステム変数は表 2-6を参照してください．

2.2.6. CaoRobot::Haltメソッド

CaoRobot クラスの Move，Execute メソッド等で行うロボット動作命令を実行した場合，Halt メソッドによりロボット動作を途中で停止させることが出来ます．

2.2.7. CaoRobot::Moveメソッド

ロボットが指定した位置へ移動します．第1引数にMoveコマンドの種類，第2引数にVARIANT配列で移動先情報，速度を指定します．以下にMoveの仕様を示します．

書式 Move <lComp:VT_I4>,<vntParam : (VT_VARIANT | VT_ARRAY)>

< lComp > : [in] コマンド番号 (VT_I4) 表 2-3参照
< vntParam > : [in] 移動先パラメータ配列 (VT_VARIANT | VT_ARRAY)
表 2-4参照

表 2-3 Move コマンド第１引数と MOV コマンド対応表

第 1 引数	コマンド	説明
1	MOVD	指定された座標位置に移動します．
2	MOVA	指定されたポイント番号のデータの位置に移動します．
3	MOVI	指定されたポイント番号のデータ量だけ現在の位置から移動します．
4	MOVF	未使用．

5	MOVL	DRCX(2 軸)コントローラのみ．指定されたポイント番号のデータの位置に直線補間で移動します．
6	MOVC	DRCX(2 軸)コントローラのみ．番号で指定したポイントを通る円弧補間移動を実行します．
7	MOVM	マトリックスの指定されたパレットワーク位置に移動します．

移動先パラメータを第 2 引数に配列で指定します．

表 2-4 Move コマンド第 2 引数パラメータ

MOV コマンド	第 2 引数パラメータ	備考
MOVD	<X 軸位置(mm):VT_R8>,<速度:VT_I4>	
MOVA	<ポイント番号:VT_I4>,<速度:VT_I4>	ポイント番号には，ポイント変数 P も使用できます．
MOVI	<ポイント番号:VT_I4>,<速度:VT_I4>	ポイント番号には，ポイント変数 P も使用できます．
MOVF	未使用	未使用
MOVL	<ポイント番号:VT_I4>,<最高速度:VT_I4>	ポイント番号には，ポイント変数 P も使用できます．
MOVC	<ポイント番号:VT_I4>,<最高速度:VT_I4>,<軌跡指定:VT_I4>	ポイント番号には，ポイント変数 P も使用できます．
MOVM	<パレットワーク位置:VT_I4>,<速度:VT_I4>	パレットワーク位置は 1~65025(255×255)の値を指定します．

使用例

・単軸コントローラでの Move 使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot("AAA")

aaa.Move 1, Array(150.5, 100)      ' MOVD コマンド 150.5mm の位置へ速度 100%で移動
aaa.Move 2, Array(10, 50)          ' MOVA コマンド ポイント番号 10 の位置へ速度 50%で移動
aaa.Move 3, Array(11, 30)          ' MOVI コマンド ポイント番号 11 のデータ量だけ現在位置
                                     から速度 30%で移動
aaa.Move 7, Array(1, 100)          ' MOVM コマンド マトリックス定義 1 の位置へ速度 100%で移動
```

・2 軸コントローラでの Move 使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot("AAA")

aaa.Move 5, Array(20, 100)     ' MOVL コマンド ポイント番号 20 の位置へ速度 100%で移動
aaa.Move 6, Array(30, 50, 0)   ' MOVC コマンド 現在位置, ポイント番号 30, ポイント番号 31 の 3 点
                               で形成される円弧軌跡上を速度 50%で移動
```

2.2.8. CaoRobot::Executeメソッド

ロボットクラスのコマンドを実行します.

Execute メソッドの引数は, コマンドを BSTR,パラメータを VARIANT 配列で指定します.

各コマンドの詳細は3コマンドリファレンスを参照してください.

書式 [<vntRet:VT_VARIANT>=]Execute(<bstrCmd:VT_BSTR>[,<vntParam:VT_VARIANT>])

< vntRet >	:	[out] コマンドの返り値 (VT_VARIANT)
< bstrCmd >	:	[in] コマンド (VT_BSTR)
< vntParam >	:	[in] パラメータ (VT_VARIANT)

使用例

```
Dim vRes As Variant
vRes = caoCtrl.Execute("SRVO", 1)  ' サーボを ON
```

2.2.9. CaoVariable::get_valueプロパティ

オブジェクトに対応している変数の値を取得します.

変数の実装状況およびデータ型は表 2-5,表 2-6を参照してください.

2.2.10. CaoVariable::put_valueプロパティ

オブジェクトに対応している変数に値を設定します.

変数の実装状況およびデータ型は表 2-5,表 2-6を参照してください.

2.3. 変数一覧

2.3.1. コントローラクラス

表 2-5 コントローラクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@VER	VT_BSTR	コントローラのバージョン情報を取得します.	○	
@CLOCK	VT_BSTR	コントローラの総起動時間を取得します.	○	
@EMG	VT_I2	非常停止の状態を取得します. 0： 非常停止解除状態, 1： 非常停止状態	○	
@MODE	VT_I2	ロボットの状態を取得します. 0： 停止状態 1：PC 通信でプログラム実行状態 2：I/O 命令でプログラム実行状態	○	
@Timeout	VT_I4	通信のタイムアウト時間(ミリ秒)を設定/取得します.	○	○

2.3.2. ロボットクラス

表 2-6 ロボットクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@POS[n]	VT_R8	ロボットの現在位置を取得します. [n]:軸番号 0:全軸, 1:X 軸, 2:Y 軸, 省略:1 軸) 軸指定は DRCX(2 軸)コントローラで使用します.	○	
@NO	VT_BSTR	現在実行中のプログラム番号を取得します. マルチタスク動作をしている場合は選択されているタスクのプログラムに関する情報となります. (例) “10/1” : No.1 が先頭プログラムで, 現在 No.10 のプログラムを実行中	○	

@SNO	VT_BSTR	現在のステップ番号を取得します．マルチタスク動作をしている場合は選択されているタスクのプログラムに関する情報となります．	○	
@TNO	VT_BSTR	現在選択されているタスクの番号を取得します．	○	
@PNO	VT_BSTR	現在選択されているポイント番号を取得します．マルチタスク動作をしている場合は選択されているタスクのプログラムに関する情報となります．	○	
@MEM	VT_BSTR	追加可能ステップ数を取得します．	○	
@ROBOT	VT_BSTR	現在設定されているロボットタイプを取得します．	○	
@PVA	VT_BSTR	ポイント変数Ｐの番号を取得します．マルチタスク動作をしている場合は選択されているタスクのプログラムに関する情報となります． 〈注〉 値保持の有効範囲 電源 OFF ⇒ 保持 プログラムリセットのかかる操作 ⇒ 0 初期化	○	
@MSEL	VT_BSTR	現在指定されているマトリックスのパレット番号を取得します．マルチタスク動作をしている場合は選択されているタスクのプログラムに関する情報となります．	○	
@SHFT	VT_BSTR	現在選択されているシフトデータを取得します．マルチタスク動作をしている場合は選択されているタスクのプログラムに関する情報となります．	○	
@SRVO[n]	VT_I2	サーボの状態を取得します． 0： サーボオフ 1： サーボオン [n]:軸番号 0:全軸，1:X 軸，2:Y 軸，省略:1 軸) 軸指定は DRCX(2 軸)コントローラで使用します．	○	
@ORG[n]	VT_I2	原点復帰の完了状態を取得します． 0： 原点未了状態 1： 原点復帰完了状態 [n]:軸番号 0:全軸，1:X 軸，2:Y 軸，省略:1 軸) 軸指定は DRCX(2 軸)コントローラで使用します．	○	

2.4. エラーコード

SR1 プロバイダでは，以下の固有エラーコードが定義されています．ORiN2 共通エラーについては，

「ORiN2 プログラミングガイド」のエラーコードの章を参照してください.

表 2-7 独自エラーコード一覧

エラー名	エラー番号	説明
E_CAOP_NO_LICENSE	0x80100000	ライセンスがありません. 追加ライセンスを購入してください.
SR1 コマンドエラー	0x8011xxxx	SR1 のコマンド実行時にエラーが発生した場合は，をxxxx の箇所にエラー番号を入れて返します. エラーコードの内容については SR1 のマニュアルを参照してください.

3. コマンドリファレンス

本章ではCaoController::Execute,CaoRobot::Executeメソッドの各コマンドについて解説します．各コマンドの詳細動作については YAMAHA ロボットコントローラ取扱説明書を参照してください．

3.1. Controllerクラス

表 3-1 CaoController::Execute コマンド一覧

コマンド	機能	
?ALM	過去に発生したアラームの履歴を取得します．	P.16
?ERR	過去に発生したエラーの履歴を取得します．	P.16
?PRM	指定パラメータを取得します．	P.17
?STP	指定したプログラムの総ステップ数を取得します．	P.17
ALMRST	アラームリセットを実行します．	P.18
SendControlCode	制御コマンドを送信します．	P.18
NativeSend	Telnet 通信によるデータの送信を行います．	P.19
NativeReceive	Telnet 通信によるデータの受信を行います．	P.19

3.1.1. CaoController::Execute(“?ALM”) コマンド

コントローラで起きたアラームを取得します．データは<番号>,<アラーム日時>,<アラーム内容>の文字列の配列で取得できます．表示数を省略して指定した場合は，最新のアラームデータを取得します．

アラーム内容の詳細は，YAMAHA ロボットコントローラ取扱説明書を参照してください．

書式　?ALM (< vntAlmParam:VT_VARIANT >)

< vntAlmParam > : [in] アラーム取得パラメータ (VT_VARIANT)

<lNo>	履歴番号:VT_I4
[<lCount>]	表示数:VT_I4

戻り値 : [out] アラーム文字列配列 (VT_BSTR | VT_ARRAY)

使用例

```
Dim aaa As Variant
aaa = caoCtrl.Execute(“?ALM”, Array(0, 2))    ‘ もっとも最近発生したアラームを 2 つ取得
```

3.1.2. CaoController::Execute(“?ERR”) コマンド

コントローラで起きたエラーを取得します．データは<番号>,<エラー日時>,<エラー内容>の文字列の配

列で取得できます．表示数を省略して指定した場合は，最新のエラーデータを取得します．

エラー内容の詳細は，YAMAHA ロボットコントローラ取扱説明書を参照してください．

書式 ?ERR (<vntErrParam:VT_VARIANT>)

< vntErrParam > : [in] エラー取得パラメータ (VT_VARIANT)

<lNo>	履歴番号:VT_I4
[<lCount>]	表示数:VT_I4

戻り値 : [out] エラー文字列配列 (VT_BSTR | VT_ARRAY)

使用例

```
Dim aaa As Variant
aaa = caoCtrl.Execute("?ERR", Array(0, 2))    ' もっとも最近発生したエラーを 2 つ取得
```

3.1.3. CaoController::Execute("?PRM") コマンド

コントローラの設定パラメータを取得します．データは<パラメータ番号>,<データ>の整数型の配列で取得できます．配列の要素 1 で指定した場合は，1 つのパラメータのデータを取得します．

パラメータの詳細は，YAMAHA ロボットコントローラ取扱説明書を参照してください．

書式 ?PRM (<vntParamData:VT_VARIANT>)

< vntParamData > : [in] パラメータデータ(VT_VARIANT)

<lBeginPrmNo>	開始パラメータ番号
[<lEndPrmNo>]	終了パラメータ番号

戻り値 : [out] パラメータ番号とデータ (VT_I4 | VT_ARRAY)

使用例

```
Dim aaa As Variant
aaa = caoCtrl.Execute("?PRM", Array(110, 114))    ' パラメータ番号 110～114 を取得
```

3.1.4. CaoController::Execute("?STP") コマンド

指定したプログラムの総ステップ数を取得します．データは整数型で取得できます．

書式 ?STP (<lProgramNo:VT_I4>)

< lProgramNo > : [in] プログラム番号 (VT_I4)
戻り値 : [out] プログラムの総ステップ数 (VT_I4)

使用例

```
Dim aaa As Variant
aaa = caoCtrl.Execute("?STP", 0) ' プログラム0番の総ステップ数を取得
```

3.1.5. CaoController::Execute("ALMRST") コマンド

コントローラで起きたアラームをリセットします．リセット可能なアラームのみリセットできます．リセット不可能なアラームについては，YAMAHA ロボットコントローラ取扱説明書を参照してください．

書式 ALMRST ()

戻り値 : なし

使用例

```
caoCtrl.Execute("ALMRST")          ' アラームをリセット
```

3.1.6. CaoController::Execute("SendControlCode") コマンド

コントローラに対して，1 バイトの制御コードを送信します．制御コードについては，YAMAHA ロボットコントローラ取扱説明書を参照してください．

書式 SendControlCode (<bytCode:VT_UI1>)

< bytCode > : [in] 制御コード (VT_UI1)

0x03	ˆC:ORG,XINC,XDEC などの中断
0x1A	ˆZ:データ送信終了
0x02	ˆB:アラームメッセージ出力停止

戻り値 : なし

使用例

```
caoCtrl.Execute("SendControlCode", &H03)   ` ^C 0x03 を送信
```

3.1.7. CaoController::Execute(“NativeSend”) コマンド

コントローラに対して，Telenet 通信によるデータを送信します．送信コマンドとパラメータを文字列で指定してください．

書式　NativeSend (<bstrNativeText>)

< bstrNativeText > : [in] 送信データ (VT_BSTR)
戻り値 : なし

使用例

```
caoCtrl.Execute(“NativeSend”, “@?ALM 0,2”)     ` @?ALM コマンドでパラメータ”0,2”を送信
```

3.1.8. CaoController::Execute(“NativeReceive”) コマンド

コントローラに対して，Telenet 通信によるデータを受信します．コントローラからの送信データを文字列で取得します．”OK c/r l/f”, ”NG c/r l/f”などは含まれません．

書式　NativeReceive ()

戻り値 : [out] コントローラからの送信データ (VT_BSTR)

使用例

```
Dim aaa As String
aaa = caoCtrl.Execute(“NativeReceive”)          ‘ コントローラからの送信データを受信
```

3.2. Robotクラス

表 3-2 CaoRobot::Execute コマンド一覧

コマンド	機能	
ORG	原点復帰します.	P. 20
SRVO	サーボの On/Off を行います.	P. 21
X+	X 軸を JOG 移動速度の 1/100 の移動量で+方向に移動します.	P. 21
X-	X 軸を JOG 移動速度の 1/100 の移動量で-方向に移動します.	P. 22
Y+	Y 軸を JOG 移動速度の 1/100 の移動量で+方向に移動します.	P. 22
Y-	Y 軸を JOG 移動速度の 1/100 の移動量で-方向に移動します.	P. 22
XINC	X 軸を JOG 移動速度で+方向に移動し続けます.	P. 23
XDEC	X 軸を JOG 移動速度で-方向に移動し続けます.	P. 23
YINC	Y 軸を JOG 移動速度で+方向に移動し続けます.	P. 24
YDEC	Y 軸を JOG 移動速度で-方向に移動し続けます.	P. 24
DRVD	指定した軸を指定された座標位置へ移動します.	P. 25
DRVA	指定した軸をポイントデータの位置へ移動します.	P. 25
DRVI	指定した軸をポイントデータの量だけ現在位置から移動します.	P. 26
ACHA	位置指定のアーチモーションを定義します.	P. 26
ACHI	距離指定のアーチモーションを定義します.	P. 27
P	ポイント変数 P を設定します.	P. 27
P+	ポイント変数 P に 1 を加算します.	P. 28
P-	ポイント変数 P に 1 を減算します.	P. 28
MAT	マトリックスを定義します.	P. 28
MSEL	Move メソッドのマトリックス移動で使用するマトリックス番号を設定します.	P. 29
SHFT	指定したポイントデータ分を位置データにシフト実行します.	P. 29
WRITE	ポイントデータの書込みを行います.	P. 30
?P	指定ポイントデータを取得します.	P. 30
?MAT	指定マトリックスデータを取得します.	P. 31

3.2.1. CaoRobot::Execute(“ORG”) コマンド

突き当て原点復帰動作を行い, 正常終了時にマシンリファレンス量を出力します.

書式　ORG ([<lAxisNo:VT_I2>])

[<lAxisNo>] : [in] 軸番号 1:X 軸, 2:Y 軸

戻り値 : [out] 原点復帰完了後のマシンリファレンス量(VT_I4)

使用例

```
Dim aaa As Object
Dim bbb As Long

Set aaa = caoCtrl.AddRobot("AAA")
bbb = aaa.Execute("ORG")          ' 原点復帰
```

3.2.2. CaoRobot::Execute("SRVO") コマンド

サーボの On/Off 制御を行います.

書式 SRVO (<lSrvoState:VT_I2>[,<lAxisNo:VT_I2>])

< lSrvoState > : [in] サーボ状態 (VT_I2)
0:サーボオフ 1:サーボオン

[<lAxisNo>] : [in] 軸番号 1:X 軸, 2:Y 軸

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot("AAA")
aaa.Execute("SRVO",1)             ' サーボを ON
```

3.2.3. CaoRobot::Execute("X+") コマンド

X 軸を+方向に次の式で示される移動量だけ+方向に移動します.

移動量(mm) = 1 × (PRM201 / 100)　　PRM201:JOG 移動速度(mm/s)

書式 X+ ()

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot("AAA")
aaa.Execute("X+")       ' X+に(PRM201)/100 だけ移動
```

3.2.4. CaoRobot::Execute(“X−”) コマンド

X 軸を+方向に次の式で示される移動量だけ-方向に移動します.

移動量(mm) = 1 × (PRM201 / 100) PRM201:JOG 移動速度(mm/s)

書式 X- ()

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“X−”)        ‘ X+に(PRM201)/100 だけ移動
```

3.2.5. CaoRobot::Execute(“Y+”) コマンド

DRCX 専用コマンド.

Y 軸を+方向に次の式で示される移動量だけ+方向に移動します.

移動量(mm) = 1 × (PRM12 / 100) PRM12:ティーチ移動データ(%)

また, ロボットが回転軸の場合, 単位は deg/sec となります.

書式 Y+ ()

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“Y+”)        ‘ Y+に(PRM12)/100 だけ移動
```

3.2.6. CaoRobot::Execute(“Y−”) コマンド

DRCX 専用コマンド.

Y 軸を-方向に次の式で示される移動量だけ-方向に移動します.

移動量(mm) = 1 × (PRM12/ 100) PRM12:ティーチ移動データ(%)

また, ロボットが回転軸の場合, 単位は deg/sec となります.

書式 Y- ()

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“Y-”)        ‘ Y-に(PRM12)/100 だけ移動
```

3.2.7. CaoRobot::Execute(“XINC”) コマンド

X 軸が+側に，PRM201(JOG 移動速度)で，制御コード(ˆC)が入力されるか，ソフトリミットに達するまで移動を続けます．

[注意]

原点未了状態ではソフトリミットは無効です．

書式 XINC ()

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“XINC”)      ‘ X+方向に JOG 移動速度で移動し続ける
```

3.2.8. CaoRobot::Execute(“XDEC”) コマンド

X 軸が-側に，PRM201(JOG 移動速度)で，制御コード(ˆC)が入力されるか，ソフトリミットに達するまで移動を続けます．

[注意]

原点未了状態ではソフトリミットは無効です．

書式 XDEC ()

戻り値 : なし

使用例

```
Dim aaa As Object
```

```
Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“XDEC”)      ‘ X-方向に JOG 移動速度で移動し続ける
```

3.2.9. CaoRobot::Execute(“YINC”) コマンド

DRCX 専用コマンド.

Y 軸が+側に, PRM201(JOG 移動速度)で, 制御コード(ˆC)が入力されるか, ソフトリミットに達するまで移動を続けます. また, ロボットが回転軸の場合, 単位は deg/sec となります.

[注意]

原点未了状態ではソフトリミットは無効です.

書式　　YINC ()

戻り値　　:　なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“YINC”)      ‘ Y+方向に JOG 移動速度で移動し続ける
```

3.2.10. CaoRobot::Execute(“YDEC”) コマンド

DRCX 専用コマンド.

Y 軸が-側に, PRM201(JOG 移動速度)で, 制御コード(ˆC)が入力されるか, ソフトリミットに達するまで移動を続けます. また, ロボットが回転軸の場合, 単位は deg/sec となります.

[注意]

原点未了状態ではソフトリミットは無効です.

書式　　YDEC ()

戻り値　　:　なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“YDEC”)      ‘ Y+方向に JOG 移動速度で移動し続ける
```

3.2.11. CaoRobot::Execute(“DRVD”) コマンド

DRCX 専用コマンド.

指定した軸を指定された座標位置へ移動します.

書式 DRVD (<vntDRVParam:VT_VARIANT | VT_ARRAY>)

< vntDRVParam > : [in] ドライブ情報 (VT_VARIANT | VT_ARRAY)

配列データ:

< vntAxis >	軸番号 1:X 軸, 2:Y 軸
<vntPosition>	移動位置 (mm) ただし, ロボットが回転軸設定の時 (deg)
< vntSpeed >	速度 1～100

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“DRVD”, Array(1,150.55,100))  ‘ X 軸が X=150.55 の位置へ速度 100%で移動する
```

3.2.12. CaoRobot::Execute(“DRVA”) コマンド

DRCX 専用コマンド.

指定した軸を指定されたポイントデータ位置へ移動します.

書式 DRVA (<vntDRVParam:VT_VARIANT | VT_ARRAY>)

< vntDRVParam > : [in] ドライブ情報 (VT_VARIANT | VT_ARRAY)

配列データ:

< vntAxis >	軸番号 1:X 軸, 2:Y 軸
<vntPointNo>	ポイント番号 0～999 ポイント変数 P も使用できます.
< vntSpeed >	速度 1～100

戻り値 : なし

使用例

```
Dim aaa As Object
```

```
Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“DRVA”, Array(1,123,100))      ‘ X 軸がポイント 123 へ速度 100%で移動する
```

3.2.13. CaoRobot::Execute(“DRVI”) コマンド

DRCX 専用コマンド.

指定した軸を指定されたポイントデータ位置量だけ現在位置から移動します.

書式　DRVI (<vntDRVParam:VT_VARIANT | VT_ARRAY>)

< vntDRVParam >　:　[in] ドライブ情報 (VT_VARIANT | VT_ARRAY)

配列データ:

< vntAxis >	軸番号 1:X 軸, 2:Y 軸
<vntPointNo>	ポイント番号 0～999 ポイント変数 P も使用できます.
< vntSpeed >	速度 1～100

戻り値　:　なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“DRVI”, Array(2,150.55,100))    ‘ Y 軸がポイント 123 のデータ量だけ現在位置から
                                            速度 100%で移動する
```

3.2.14. CaoRobot::Execute(“ACHA”) コマンド

DRCX 専用コマンド.

位置指定(原点基準の絶対位置)のアーチモーションを定義します.

書式　ACHA (<vntACHParam:VT_VARIANT | VT_ARRAY>)

< vntACHParam >　:　[in] アーチ情報 (VT_VARIANT | VT_ARRAY)

配列データ:

< vntAxis >	軸番号 1:X 軸, 2:Y 軸
< vntPosition >	指定位置 -9999～9999 (mm)

戻り値　:　なし

使用例

```
Dim aaa As Object
```

```
Set aaa = caoCtrl.AddRobot("AAA")
aaa.Execute("ACHA", Array(2,10))          ' Y=10.00 のポイントまで戻るアーチモーション定義
```

3.2.15. CaoRobot::Execute("ACHI") コマンド

DRCX 専用コマンド.

位置指定(現在位置基準の相対位置)のアーチモーションを定義します.

書式　ACHI (<vntACHParam:VT_VARIANT | VT_ARRAY>)

< vntACHParam > : [in] アーチ情報 (VT_VARIANT | VT_ARRAY)

配列データ:

< vntAxis >	軸番号 1:X 軸, 2:Y 軸
< vntPosition >	指定距離 -9999～9999 (mm)

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot("AAA")
aaa.Execute("ACHI", Array(2,-100))          ' Y=-100.00 の距離だけ戻るアーチモーション定義
```

3.2.16. CaoRobot::Execute("P") コマンド

ポイント変数 P を設定します. ポイント番号は 0～999 まで指定できます.

[注意]

コントローラの電源を OFF してもポイント変数の内容は保持されますが, プログラムリセットがかかる操作を行った場合, ポイント変数は 0 に初期化されます.

書式　P (<lPNo:VT_I4>)

< lPNo > : [in] ポイント番号 (VT_I4) 0～999 までのポイント番号

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot("AAA")
aaa.Execute("P", 100)   ' ポイント変数 P を 100 に設定
```

3.2.17. CaoRobot::Execute(“P+”) コマンド

ポイント変数 P に 1 を加算します.

書式　P+ ()

戻り値　:　なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“P”, 100)   ‘ ポイント変数 P を 100 に設定
aaa.Execute(“P+”)       ‘ ポイント変数 P は 101 (P←P+1)
```

3.2.18. CaoRobot::Execute(“P-”) コマンド

ポイント変数 P に 1 を減算します.

書式　P- ()

戻り値　:　なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“P”, 100)   ‘ ポイント変数 P を 100 に設定
aaa.Execute(“P-”)       ‘ ポイント変数 P は 99 (P←P-1)
```

3.2.19. CaoRobot::Execute(“MAT”) コマンド

マトリックスを定義します. 行数, 列数はそれぞれ 1～255 までの値を指定できます. パレット番号は固有値 0～31 を指定できます.

書式　MAT (<vntMatParam:VT_I4 | VT_ARRAY>)

< vntMatParam >　:　[in] マトリックス定義情報 (VT_I4 | VT_ARRAY)

配列データ:

< lRowCount >	行数 1～255
< lColCount >	列数 1～255
<lPalltNo>	パレット番号 0～31

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“MAT”, Array(5,2,1))          ‘5×2 のマトリックスを１番目のパレットに設定
```

3.2.20. CaoRobot::Execute(“MSEL”) コマンド

書式 MSEL (<lPalletNo:VT_I4>)

< lPalletNo > : [in] パレット番号 (VT_I4)
マトリックス区別用の 0～31 までのパレット番号

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“MAT”, Array(5,2,1))          ‘5×2 のマトリックスを１番目のパレットに設定
aaa.Execute(“MSEL”, 1)                    ‘１番目のパレットを選択
aaa.Move 7, Array(1,100)                  ‘１番目のパレット位置に速度 100%で移動
```

3.2.21. CaoRobot::Execute(“SHFT”) コマンド

指定したポイントデータ分を位置データにシフト実行します.

再度 SHFT 文が実行されるかプログラムリセットがかかるまで有効です.

書式 SHFT (<lPointNo:VT_I4>)

< lPointNo > : [in] ポイント番号 (VT_I4) 0～999 までのポイント番号

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“SHFT”, 1)          ‘ ポイント番号 1 のデータ量シフト設定
```

3.2.22. CaoRobot::Execute(“WRITE”) コマンド

ポイントデータを書込みます．ポイントデータの書込みは文字列の配列を指定します．書込むポイントは連番でなくても書込みを行うことができます．

書式　WRITE (<vntPointData:VT_BSTR | VT_ARRAY>)

< vntPointData > : [in] ポイント設定文字列 (VT_BSTR | VT_ARRAY)

Px=<位置データ>	x:ポイント番号 位置データは VT_R8

戻り値 : なし

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“WRITE”, Array(“P0=0.00”,”P1=350.00”,”P254=-0.27”)) ‘ ポイント書込み
```

3.2.23. CaoRobot::Execute(“?P”) コマンド

指定ポイント番号のポイントデータを取得します．第 1 引数に取得するポイント番号を配列で指定します．1 ポイントのみ取得したい場合は，1 要素の配列を指定します．未登録のポイント番号のポイントデータはスキップされます．

書式 ?P (<vntPointData:VT_I4 | VT_ARRAY>)

< vntPointData > : [in] ポイント番号 (VT_I4 | VT_ARRAY)

<lBeginPointNo>	ポイント番号:VT_I4
[<lEndPointNo>]	ポイント番号:VT_I4

戻り値 : [out] ポイントデータ (VT_BSTR |VT _ARRAY)
P(lBeginPointNo)=位置データ, ...
, P(lEndPointNo)=位置データの配列

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“?P”, Array(15,24))          ‘ ポイント番号 15 から 24 のデータを取得
```

3.2.24. CaoRobot::Execute(“?MAT”) コマンド

マトリックス定義内容を取得します．マトリックス区別用の 0～31 の固有番号を指定します．

書式 ?MAT (<lMatNo:VT_I4>)

< lMatNo > : [in] マトリックス番号 (VT_I4)

戻り値 : [out] マトリックス定義データ (VT_I4 | VT_ARRAY)

行数	VT_I4
列数	VT_I4

使用例

```
Dim aaa As Object

Set aaa = caoCtrl.AddRobot(“AAA”)
aaa.Execute(“?MAT”, 1)          ‘ パレット番号 1 のデータを取得
```

付録A. SR1(DRCX)コマンド対応表

Appendix A.1. コントローラクラス

表 A-1 CaoController::Execute メソッド−SR1 コマンド対応表

コマンド名	SR1 コマンド
?ALM	?ALM
?ERR	?ERR
?PRM	?PRM
?STP	?STP
ALMRST	ALMRST
SendControlCode	-
NativeSend	-
NativeReceive	-

表 A-2 CaoController の変数オブジェクト−SR1 コマンド対応表

変数名	SR1 コマンド
@Timeout	-
@VER	?VER
@CLOCK	?CLOCK
@EMG	?EMG
@MODE	?MODE

Appendix A.2. ロボットクラス

表 A-3 CaoRobot::Execute メソッド−SR1 コマンド対応表

コマンド名	SR1 コマンド
ORG	ORG
SRVO	SRVO
X+	X+
X-	X-
Y+	Y+ (DRCX)
Y-	Y- (DRCX)
XINC	XINC

XDEC	XDEC
YINC	YINC (DRCX)
YDEC	YDEC (DRCX)
P	P
?P	?P
P+	P-
MAT	MAT
?MAT	?MAT
MSEL	MSEL
SHFT	SHFT
DRVD	DRVD (DRCX)
DRVA	DRVA (DRCX)
DRVI	DRVI (DRCX)
ACHA	ACHA (DRCX)
ACHI	ACHI (DRCX)
WRITE	WRITE PNT

表 A-4 CaoRobot::Execute 以外のメソッド–SR1 コマンド対応表

メソッド	SR1 コマンド
Move	MOVD
	MOVA
	MOVI
	MOVF(未使用)
	MOVL (DRCX)
	MOVC (DRCX)
	MOVM
Halt	^C

表 A-5 CaoRobot の変数オブジェクト–SR1 コマンド対応表

変数名	SR1 コマンド
@POS	?POS
@NO	?NO
@SNO	?SNO

@TNO	?TNO
@PNO	?PNO
@MEM	?MEM
@ROBOT	?ROBOT
@PVA	?PVA
@MSEL	?MSEL
@SHIFT	?SHIFT
@SRVO	?SRVO
@ORG	?ORG